Descriptive analysis

R for Stata Users

Luiza Andrade, Leonardo Viotti & Rob Marty

April 2019

Outline

Introduction

- Descriptive statistics are used to represent the basic features of data. When we talk about descriptive analysis, it usually means that we're not making any assumptions, and we're not using probability theory to infer anything beyond the immediate data.
- This session is mostly focused on how to implement descriptive analysis in R.
- We will not go in depth into these concepts, but you can find some useful references at the end of this presentation.

Introduction

This session will cover two topics:

1. Quick ways to extract summary information from your data.
2. How to use this information to create tables.
3. How to export these tables to LaTeX and Excel.

Outline

Getting started

First, let's load the data that is going to be used in the training.

Load the data

```
# Replace with the path to your dime-r-training folder
projectFolder <- file.path("YOUR/FOLDER/PATH/HERE")

dataWorkFolder    <- file.path(projectFolder,
                               "DataWork")
finalData         <- file.path(dataWorkFolder,
                               "DataSets", "Final")
rawOutput         <- file.path(dataWorkFolder,
                               "Output","Raw")

whr <- read.csv(file.path(finalData,"whr_panel.csv"),
                header = T,
                stringsAsFactors = F)
```

Getting started

Before starting, lets install the packages we'll use in this session since it might take a while.

Install today's packages

```
install.packages(c("stargazer",
                   "tidyverse",
                   "openxlsx"),
                 dependencies = T)
```

Outline

Quick summary statistics

`summary(x, digits)`

Equivalent to Stata's `summarize`, displays summary statistics. Its arguments are:

- **x:** the object you want to summarize, usually a vector or data frame
- **digits:** the number of decimal digits to be displayed

Exercise 1

Use the `summary()` function to display summary statistics for the `whr` data frame.

Quick summary statistics

```r
# Summary statistics
summary(whr)
```

```
##   country_code     country             region               year
##  Min.   :  4.0   Length:470         Length:470         Min.   :2015
##  1st Qu.:203.0   Class :character   Class :character   1st Qu.:2015
##  Median :418.0   Mode  :character   Mode  :character   Median :2016
##  Mean   :429.6                                         Mean   :2016
##  3rd Qu.:646.0                                         3rd Qu.:2017
##  Max.   :894.0                                         Max.   :2017
##  NA's   :5
##    happy_rank      happy_score        gdp_pc
##  Min.   :  1.00   Min.   :2.693   Min.   :0.0000
##  1st Qu.: 40.00   1st Qu.:4.509   1st Qu.:0.6053
##  Median : 79.00   Median :5.282   Median :0.9954
##  Mean   : 78.83   Mean   :5.371   Mean   :0.9278
##  3rd Qu.:118.00   3rd Qu.:6.234   3rd Qu.:1.2524
##  Max.   :158.00   Max.   :7.587   Max.   :1.8708
##
```

Quick summary statistics

`table()`

Equivalent to `tabulate` in Stata, creates a frequency table. Its main arguments are vectors to be tabulated.

Exercise 2

Use the `table()` function to display frequency tables for:

1. The variable `year` in the `whr` data frame
2. The variables `region` and `year` in the `whr` data frame, simultaneously

Quick summary statistics

```
# Year of data collection
table(whr$year)
```

```
##
## 2015 2016 2017
##  158  157  155
```

Quick summary statistics

```r
# Number of countries per region per year
table(whr$region, whr$year)
```

```
##
##                                  2015 2016 2017
##   Australia and New Zealand         2    2    2
##   Central and Eastern Europe       29   29   29
##   Eastern Asia                      6    6    6
##   Latin America and Caribbean      22   24   22
##   Middle East and Northern Africa  20   19   19
##   North America                     2    2    2
##   Southeastern Asia                 9    9    8
##   Southern Asia                     7    7    7
##   Sub-Saharan Africa               40   38   39
##   Western Europe                   21   21   21
```

Quick summary statistics

Bonus Exercise:

Use the `table()` function to display a frequency table for the number of countries **above the average happiness** per region in 2017.

1. Create another data.frame called `whr17` with only 2017 observations
2. Use the `table()` function to tabulate a the `region` variable and a boolean vector.

- TIP: Using the condition directly in the function or creating a separate vector will yield the exact same results.

Quick summary statistics

```r
# Restrict to only 2017 obs
whr17 <- whr[whr$year == 2017, ]

# table with a bolean vector
table(whr17$region,
      whr17$happy_rank > mean(whr17$happy_rank)) # Who is above average
```

```
##
##                                  FALSE TRUE
##   Australia and New Zealand          2    0
##   Central and Eastern Europe        18   11
##   Eastern Asia                       4    2
##   Latin America and Caribbean       18    4
##   Middle East and Northern Africa   10    9
##   North America                      2    0
##   Southeastern Asia                  4    4
##   Southern Asia                      0    7
##   Sub-Saharan Africa                 1   38
##   Western Europe                    19    2
```

Outline

Descriptives tables

We can also use the `stargazer()` function to quickly display a nice-looking descriptives table.

Stargazer was originally developed to export beautiful regression tables to LaTeX or html, but it also allows you to generate summary statistics.

Descriptives tables

The `stargazer()` function accepts **a lot** of arguments, most of which are beyond the scope of this session. Here are the arguments you'll need for this specific table

`stargazer()`

- **x:** the object you want to summarize – in this case a vector or data frame
- **type:** the output format – "text" to just display, "latex" (the default) to save as a LaTeX table, and "html" for, yes, html
- **digits:** the number of decimal digits to be displayed

Descriptives tables

Exercise 3 - `stargazer()` summary statistics table

Use the `stargazer()` function to display (on your R console) summary statistics for the variables in the *whr* data frame.

- TIP: Set the type argument to "text".

Descriptives tables

```
# A descriptive table with stargazer
stargazer(whr,
          digits = 1,
          type = "text")
```

```
##
## ===================================================================
## Statistic        N    Mean    St. Dev.  Min  Pctl(25) Pctl(75)  Max
## -------------------------------------------------------------------
## country_code    465  429.6    255.4    4.0   203.0    646.0   894.0
## year            470 2,016.0    0.8    2,015 2,015    2,017   2,017
## happy_rank      470  78.8      45.3     1      40       118     158
## happy_score     470   5.4       1.1     2.7    4.5      6.2     7.6
## gdp_pc          470   0.9       0.4     0.0    0.6      1.3     1.9
## family          470   1.0       0.3     0.0    0.8      1.2     1.6
## health          470   0.6       0.2     0.0    0.4      0.8     1.0
## freedom         470   0.4       0.2     0.0    0.3      0.5     0.7
## trust_gov_corr  470   0.1       0.1     0.0    0.1      0.2     0.6
## generosity      470   0.2       0.1     0.0    0.2      0.3     0.8
## dystopia_res    470   2.1       0.6     0.3    1.7      2.5     3.8
## -------------------------------------------------------------------
```

Outline

Export tables to LaTeX

To export the table to LaTeX, we will use a couple of additional arguments of the `stargazer()` function:

- **out:** where to save the table, i.e. the file path including the file name
- **covariate.labels:** a vector of variable labels

Export tables to LaTeX

But first, let's pick a few variables of interest in the `whr` data set so the table fits in these slides.

Exercise 3

1. Create a vector called `covariates` containing the string names of the variables you want to keep: `happy_score`, `gdp_pc`, `family`, and `trust_gov_corr`.
2. Use this vector to subset the whr data-set to contain only these variables. Call the new data frame `whr_simp`.

- TIP: You can use column names indexing or the `select()` function of the `dplyr` package

Export tables to LaTeX

```
# Vector with covariates to be kept
covariates <- c("happy_score",
                "gdp_pc",
                "freedom",
                "trust_gov_corr")

# subset whr
whr_simp <- whr[, covariates]
```

Export tables to LaTeX

Exercise 4

Now use the stargazer function to export the `whr_simp`:

1. Create a vector `cov_labels` containing labels for the `happy_score`, `gdp_pc`, `freedom` and `trust_gov_corr` variables.
2. Set `whr_simp` as the `x` argument this time
3. Set the `covariate.labels` argument as the vector you just created
4. Set the `out` argument to save the table in the `rawOutput` folder we defined

Export tables to LaTeX

```r
# Set labels
cov_labels <- c("Happy score", "GDP per capita",
                "Freedom", "Trust in gornment and currption")

# Save table to latex
stargazer(whr_simp,
          # Formatting
          covariate.labels = cov_labels,
          summary.stat = c("n", "mean", "sd", "min", "max"), # Chose which stats to comopute
          digits = 2,
          # You can directly export with the out argument
          out = file.path(rawOutput,"desc_table.tex"))
```

Table 1:

Statistic	N	Mean	St. Dev.	Min	Max
Happy score	470	5.37	1.14	2.69	7.59
GDP per capita	470	0.93	0.42	0.00	1.87
Freedom	470	0.40	0.15	0.00	0.67
Trust in gornment and currption	470	0.13	0.11	0.00	0.55

Outline

Descriptives tables - Create tables from scratch

In R, it is relatively straightforward to construct any table you can think of by manipulating objects.

There are multiple ways to do this, but We will construct a simple table using two functions:

- `aggregate()` - Similar to collapse in Stata, it can compute statistics of a variable based on the values of other variable.
- `spread()` - Reshapes data sets from long to wide. [1]

[1] `gather()` Reshapes data sets from wide to long, both are from `tidyverse`. There are other ways to reshape data, but these are becoming the standard in R.

Descriptives tables - Create tables from scratch

`aggregate(formula, data, FUN, ...,)`:

- **formula:** a formula, such as y ~ x or cbind(y1, y2) ~ x1 + x2, where the y variables are numeric data to be split into groups according to the grouping x variables
- **data:** a data frame (or list) from which the variables in formula should be take
- **FUN:** a function to compute statistics
- **...:** further arguments passed to or used by FUN

Descriptives tables - Create tables from scratch

Exercise 5

Use the `aggregate()` function to create a data frame called `happy_table` with the mean of `happy_score` per year and region.

```
# Aggregate happy_score by year and region
happy_table <-
  aggregate(happy_score ~ year + region,
            data = whr,
            FUN = mean)
```

For comparison, here's how you'd do it in Stata: `collapse (mean) happy_score, by(region year)`

Descriptives tables - Create tables from scratch

```
print(happy_table)
```

```
##    year                          region happy_score
## 1  2015       Australia and New Zealand    7.285000
## 2  2016       Australia and New Zealand    7.323500
## 3  2017       Australia and New Zealand    7.299000
## 4  2015      Central and Eastern Europe    5.332931
## 5  2016      Central and Eastern Europe    5.370690
## 6  2017      Central and Eastern Europe    5.409931
## 7  2015                    Eastern Asia    5.626167
## 8  2016                    Eastern Asia    5.624167
## 9  2017                    Eastern Asia    5.646667
## 10 2015     Latin America and Caribbean    6.144682
## 11 2016     Latin America and Caribbean    6.101750
## 12 2017     Latin America and Caribbean    5.957818
## 13 2015 Middle East and Northern Africa    5.406900
## 14 2016 Middle East and Northern Africa    5.386053
## 15 2017 Middle East and Northern Africa    5.369684
## 16 2015                   North America    7.273000
## 17 2016                   North America    7.254000
## 18 2017                   North America    7.154500
## 19 2015                Southeastern Asia    5.317444
```

Descriptives tables - Create tables from scratch

`spread(data, key, value)`:

- **data**: a data frame
- **key**: the variables that identify the group in the wide data set
- **value**: the variable in long format that has multiple records from the same group or individual

Descriptives tables - Create tables from scratch

Exercise 6

Use the `spread` function to make the `happy_table` data frame wide in the `year` variable.

```
# Reshape into wide on year
happy_table <-
  spread(happy_table,
         key = year,
         value = happy_score)
```

For comparison, here's how you'd do it in Stata: `reshape wide happy_score, i(region) j(year)`

Descriptives tables - Create tables from scratch

```
print(happy_table)
```

```
##                              region     2015     2016     2017
## 1          Australia and New Zealand 7.285000 7.323500 7.299000
## 2        Central and Eastern Europe 5.332931 5.370690 5.409931
## 3                       Eastern Asia 5.626167 5.624167 5.646667
## 4       Latin America and Caribbean 6.144682 6.101750 5.957818
## 5  Middle East and Northern Africa 5.406900 5.386053 5.369684
## 6                     North America 7.273000 7.254000 7.154500
## 7                 Southeastern Asia 5.317444 5.338889 5.444875
## 8                     Southern Asia 4.580857 4.563286 4.628429
## 9                Sub-Saharan Africa 4.202800 4.136421 4.111949
## 10                   Western Europe 6.689619 6.685667 6.703714
```

Descriptives tables - Create tables from scratch

With a data frame as input, `stargazer` by default tries to summarize it. So, to export this table we must specify one additional argument: `summary = F`.

Exercise 7

Print the `happy_table` table you created in exercise 6 using `stargazer`. If you want, you can also save it using the `out` option.

Descriptives tables - Create tables from scratch

```
# Create table
stargazer(happy_table,
          summary = F,
          # Exporting
          out = file.path(...),
          # Formatting:
          title = "Happy table",
          digits = 1,
          rownames = F)
```

Descriptives tables - Create tables from scratch

Table 2: Happy table

region	2015	2016	2017
Australia and New Zealand	7.3	7.3	7.3
Central and Eastern Europe	5.3	5.4	5.4
Eastern Asia	5.6	5.6	5.6
Latin America and Caribbean	6.1	6.1	6.0
Middle East and Northern Africa	5.4	5.4	5.4
North America	7.3	7.3	7.2
Southeastern Asia	5.3	5.3	5.4
Southern Asia	4.6	4.6	4.6
Sub-Saharan Africa	4.2	4.1	4.1
Western Europe	6.7	6.7	6.7

Descriptives tables - Create tables from scratch

Challenge exercise

Ok, but what if we want to create something very specific, different from the output of those two functions? Something like this:

Descriptives tables - Create tables from scratch

Challenge exercise

Table 3: Happiness score by world region

Region		2015	2016	2017
Australia and New Zealand	Mean	7.285	7.323	7.299
	N	2	2	2
Central and Eastern Europe	Mean	5.333	5.371	5.410
	N	29	29	29
Eastern Asia	Mean	5.626	5.624	5.647
	N	6	6	6
Latin America and Caribbean	Mean	6.145	6.102	5.958
	N	22	24	22
Middle East and Northern Africa	Mean	5.407	5.386	5.370
	N	20	19	19
North America	Mean	7.273	7.254	7.155
	N	2	2	2
Southeastern Asia	Mean	5.317	5.339	5.445
	N	9	9	8
Southern Asia	Mean	4.581	4.563	4.628
	N	7	7	7
Sub-Saharan Africa	Mean	4.203	4.136	4.112
	N	40	38	39
Western Europe	Mean	6.690	6.686	6.704
	N	21	21	21

Exercise 8: Try to replicate the table in the previous slide

There are multiple ways to do this. Here are two painful but straightforward approaches that you get extra points if you avoid:

1. Write string objects with latex code and combine them.
2. Appending vectors of with the desired stats for each region.

Here are a few tips if you chose to use `aggregate()` and `spread()`:

1. When using `aggregate`, the order of the right-hand-side variables affects the order of the columns.
2. The order of the columns affects the order of observations after you reshape.

Outline

Export tables to Excel

There are several ways to export R objects to Excel. We will use here the the `write.xslx()` function of the `openxlsx` package.

It takes a matrix or data frame object as input and saves it as a `.xlsx` file.

`write.xslx()` is one of the most common functions, but there are many other functions that allow you to export formatted tables to Microsoft Excel, Word or PowerPoint. Here are some examples:

- ReporteRs
- Flextable
- r2excel (only available in GitHub).

Export tables to Excel

`write.xlsx(x, file, row.names = TRUE, col.names ...)`

- **x**: the object to be written
- **file**: where to save the table, i.e., the file path including the file name
- **row.names**: a logical value indicating whether the row names of x are to be written along with x

Export tables to Excel

Exercise 9

Use the `write.xlsx()` function to save the `happy_table` you table created in Exercise 4 into an xlsx file.

1. Set `x` argument as *happy_table*.
2. Set `file` as the folder path to your output folder including a name for the file plus ".xlsx"

TIP:

- Use the help function to check syntax if needed

Export tables to Excel

```
write.xlsx(happy_table,
           file = file.path(rawOutput, "happy_table.xlsx"))
```

	A	B	C	D
1	region	2015	2016	2017
2	Australia and New Zealand	7.285	7.3235	7.299
3	Central and Eastern Europe	5.332931	5.37069	5.409931
4	Eastern Asia	5.626167	5.624167	5.646667
5	Latin America and Caribbean	6.144682	6.10175	5.957818
6	Middle East and Northern Africa	5.4069	5.386053	5.369684
7	North America	7.273	7.254	7.1545
8	Southeastern Asia	5.317444	5.338889	5.444875
9	Southern Asia	4.580857	4.563286	4.628429
10	Sub-Saharan Africa	4.2028	4.136421	4.111949
11	Western Europe	6.689619	6.685667	6.703714

Outline

Export regression table

Warning:

This is a session on **descriptive** analysis, so regressions are beyond its scope.

But since you'll probably ask, here's how you

```
# Run a Regression
reg1 <- lm(Sepal.Length ~ Sepal.Width,
           data = iris)
```

Export regression table

```
# Export a regression table

depvar_label <- "Sepal Length"
covar_labels <- "Sepal Width"

stargazer(reg1,
          title = "Regression table",
          dep.var.labels = depvar_label,
          covariate.labels = covar_labels,
          digits = 2,
          # out = "..."
          header = F)
```

Export regression table

Table 4: Regression table

	Dependent variable:
	Sepal Length
Sepal Width	−0.22
	(0.16)
Constant	6.53***
	(0.48)
Observations	150
R^2	0.01
Adjusted R^2	0.01
Residual Std. Error	0.83 (df = 148)
F Statistic	2.07 (df = 1; 148)
Note:	*p<0.1; **p<0.05; ***p<0.01

Export regression table

```r
# Regression 1
reg1 <- lm(Sepal.Length ~ Sepal.Width,
           data = iris)

# Reg with two indep vars
reg2 <- lm(Sepal.Length ~ Sepal.Width + Petal.Length,
           data = iris)

# Reg with two indep vars and species FE
reg3 <- lm(Sepal.Length ~ Sepal.Width + Petal.Length + factor(Species),
           data = iris)
```

Export regression table

```
depvar_label <- "Sepal Length"
covar_labels <- c("Sepal Width",
                  "Petal Length")
#Table
stargazer(reg1,
          reg2,
          reg3,
          font.size = "tiny",
          title = "Regression table",
          keep = c("Sepal.Width",  "Petal.Length"),
          dep.var.labels = depvar_label,
          covariate.labels = covar_labels,
          add.lines = list(
            c("Species FE", "No", "No", "Yes")
            ),
          omit.stat = c("ser"),
          digits = 2,
          header = F)
```

Export regression table

Table 5: Regression table

	Dependent variable:		
	Sepal Length		
	(1)	(2)	(3)
Sepal Width	−0.22	0.60***	0.43***
	(0.16)	(0.07)	(0.08)
Petal Length		0.47***	0.78***
		(0.02)	(0.06)
Species FE	No	No	Yes
Observations	150	150	150
R^2	0.01	0.84	0.86
Adjusted R^2	0.01	0.84	0.86
F Statistic	2.07 (df = 1; 148)	386.39*** (df = 2; 147)	228.95*** (df = 4; 145)
Note:			*p<0.1; **p<0.05; ***p<0.01

The end

Thank you!

Outline

References and recommendations

- Johns Hopkins Exploratory Data Analysis at Coursera: https://www.coursera.org/learn/exploratory-data-analysis
- Udacity's Data Analysis with R: https://www.udacity.com/course/data-analysis-with-r--ud651
- Jake Russ stargazer cheat sheet: https://www.jakeruss.com/cheatsheets/stargazer/

References and recommendations

Since we talked about LaTeX so much...

- DIME LaTeX templates and trainings: https://github.com/worldbank/DIME-LaTeX-Templates
- All you need to know about LaTeX: https://en.wikibooks.org/wiki/LaTeX

Outline

Appendix - Save your data to .csv

To export our data in .csv format we can use the `write.csv()` function. There other ways, but this is often the most straightforward.

Here's the basic syntax:

```
write.csv(x, file = "", sep = ",", row.names = TRUE)
```

- **x**: the object to be written
- **file**: where to save the table, i.e., the file path including the file name
- **sep**: the field separator of the csv, Excel's default is comma
- **row.names**: either a logical value indicating whether the row names of x are to be written along with x, or a character vector of row names to be written

Appendix - Save your data to .csv

You can write the following code:

```
write.csv(whr,
          file = file.path(...,"whr.csv"),
          row.names = F)
```

It is important to specify the `row.names` as `FALSE` since the function default is `TRUE`. There are situations when saving row names might make sense, but normally that's nor the case for data.frames.

Appendix - Formulas

Formulas are a way of describing a relationship between variables or objects. They work as inputs for several functions, notably regression functions.

We can create formulas by using the formula function

```
# or Formula function yield same results
formula1 <- formula(y ~ x1 + x2)
formula1
```

```
## y ~ x1 + x2
```

Appendix - Formulas

The most basic structure of a formula is actually just the tilde symbol ~ and at least one right-hand variable.

You can also covert strings to create formulas

```
# or Formula function yield same results
formula2 <- as.formula("~ x1")
formula2
```

```
## ~x1
```

Appendix - Formulas

Note that values that assigned to the symbols in the formula are not accessed when the formula is created.

Alternatively, if you write an expression containing a tilde R already understands it as a formula.

Just using the tilde

```
formula3 <- y ~ x1 + z1
formula3
```

```
## y ~ x1 + z1
```